Comput. Math. Applic. Vol. 14, No. 8, pp. 629–644, 1987
Printed in Great Britain.

0097-4943/87 $3.00 + 0.00

FAST AND STABLE RECURSIVE ALGORITHMS FOR CONTINUOUS-TIME AND DISCRETE-TIME MODEL CONVERSIONS

L. S. SHIEH, S. R. LIAN and C. B. PARK
Department of Electrical Engineering, University of Houston, University Park, Houston, TX 77004, U.S.A.

N. P. COLEMAN
U.S. Army Armament Center, Dover, NJ 07801, U.S.A.

(*Received* 8 *July* 1987)

Communicated by E. Y. Rodin

Abstract—Based on the Newton–Raphson method, this paper presents recursive algorithms that are rapidly convergent and more stable for modeling the equivalent continuous-time (discrete-time) model from the available discrete-time (continuous-time) model for a fixed sampling period. The newly developed recursive algorithms relax the constraints imposed upon the existing model conversion algorithms, and, thus, enhance the applications of microprocessors and associated microelectronics to digital control systems. A practical example is presented to demonstrate the effectiveness of the proposed procedures.

1. INTRODUCTION

A practical dynamic system is often described by a sampled-data system which is a collection of continuous-time and discrete-time models with their own characteristics. For simplicity in analysis and design the continuous-time (discrete-time) subsystems are converted into equivalent discrete-time (continuous-time) subsystems so that the original sampled-data system can be completely analyzed and designed in the discrete-time (continuous-time) domain via the well-developed techniques. Hence, the conversion of the continuous-time (discrete-time) model to the equivalent discrete-time (continuous-time) model is often required. Advances in industrial electronics and computer technologies have made a dramatic extension of the possibilities of control algorithms to be implemented via these high performances, low cost microprocessors and associated microelectronics; however, microprocessors are slow digital machines, and are usually equipped with a small wordlength. For effective use of microprocessors in digital control systems, the selection of a suitable sampling time and the development of recursive algorithms, which possess fast convergence rate and numerical stability to offset the slow computational speed and round-off errors, become an important part of any digital control system. The choice of the sampling time for the identification and model conversion of control systems has been discussed in Refs [1, 2]. For example, a rule of thumb for identifying a continuous-time system from samples of input/output data has been suggested in Ref. [3], which states that the sampling interval T should be chosen in such a way that $|\lambda_m T| \leqslant 0.5$, where λ_m is the magnitude of the largest eigenvalue of the continuous-time model. Without knowing the largest eigenvalue of the continuous-time system to be identified, a simple procedure for obtaining a good choice of T has been proposed in Ref. [4]. For critical dynamic systems, such as the power systems [5] and missile systems [6, 7] which contain large and small eigenvalues, a very small sampling period T is usually required for the identification and control of the systems. However, for the use of the identified model in digital control systems, the sampling period T cannot be chosen as small as we would like due to the consideration of the execution time of a particular microprocessor and the round-off errors caused by a small wordlength of the microprocessors. In other words, the choice of the sampling period T depends upon not only the dynamic property of each subsystem but also the constraints of the microprocessors used for digital control systems. In all, the development of a rapidly convergent and numerically stable recursive algorithm with a *fixed* sampling period for the model conversions

is necessary in order to enhance the applications of microprocessors and associated microelectronics to digital control systems.

Based on the Newton–Raphson method, this paper develops rapidly convergent and more stable recursive algorithms with a fixed sampling period for continuous-time and discrete-time model conversions. The proposed algorithms enable us to reduce the constraints imposed upon the existing model conversion techniques and, thus, enhance the applications of microprocessors and associated microelectronics to digital control systems. In Section 2, we review some non-recursive algorithms for finding the equivalent continuous-time (discrete-time) model from an available discrete-time (continuous-time) system. In Section 3, we develop a rapidly convergent and numerically stable recursive algorithm with a fixed sampling period T for modelling continuous-time model from a discrete-time model. In Section 4, we also develop a rapidly convergent and numerically stable recursive algorithm with a fixed sampling period for finding the discrete-time model from a continuous-time system. A practical example is given in Section 5 and the results are summarized in Section 6.

2. REVIEW OF SOME NON-RECURSIVE ALGORITHMS FOR MODEL CONVERSIONS

There are many model conversion techniques [1, 2, 8–12] available in the literature. The highly recommended techniques are summarized as follows.

(a) The scaling, squaring, continued-fraction method for finding a discrete-time model from a continuous-time model [8, 10]

Let the continuous-time system be

$$\dot{X}(t) = AX(t) + Bu(t); \quad X(0), \tag{1}$$

where $X(t)$ and $u(t)$ are the $n \times 1$ state vector and the $m \times 1$ input vector, respectively, and the A and B are constant matrices of appropriate dimensions. If we approximate $u(t)$ as a piecewise input function

$$u(t) = u(kT) = u(k) \quad \text{for} \quad kT \leqslant t < (k+1)T \tag{2}$$

where the T is the sampling period with $T < \pi/\omega_s$ where the ω_s is the highest frequency component in equation (1), then we can write the equivalent discrete-time model as

$$X(k+1) = GX(k) + Hu(k); \quad X(0), \tag{3a}$$

where

$$G = \exp(AT) \tag{3b}$$

and

$$H = [G - I_n]A^{-1}B. \tag{3c}$$

The matrix I_n denotes an $n \times n$ identity matrix and the vector $X(k) = X(kT) = X(t)$ as $t = kT$. The constraint of the sampling period $T(< \pi/\omega_s)$ in equation (2) implies that the obtained discrete-time system matrix G is a nonsingular matrix without any negative real eigenvalues. The matrices G and H can be determined exactly from the matrices A and B using the eigenvalues and eigenvectors approaches [8]. However, for simplicity, in computations without explicitly involving the eigenvalues and eigenvectors of the matrix A which may be defective, the approximant of the matrix-valued function in equation in (3b), $\exp(AT)$, is obtained for modelling the matrix G. The most commonly used method for determining the approximant is the direct truncation of the following infinite series:

$$G = \exp(AT) = I_n + AT + \frac{1}{2!}(AT)^2 + \cdots = \sum_{j=0}^{\infty} \frac{1}{j!}(AT)^j. \tag{4}$$

The truncation of the above series results in a good approximant if $T \ll 1$ and A is not a stiff matrix. A stiff matrix is defined as a matrix which contains both large and small eigenvalues. The analysis

of the truncation error for the series approximation in equation (4) has been discussed in Ref. [13], while the round-off errors have been studied in Ref. [14]. For matching the sampling period of the existing discrete-time subsystems in the original sampled-data system so that the modified sampled-data system has a *uniformly* sampled rate, the sampling period T in equation (4) can not be arbitrarily chosen. For a *fixed* sampling period, a popular method for determining the matrix G from $\exp(AT)$ is the scaling, squaring continued-fraction method [8] as follows:

$$G = (\exp(AT/q))^q \tag{5a}$$

$$\simeq [I_n + AT/q]^q \triangleq G_{2,q} \tag{5b}$$

$$\simeq \{[I_n - \tfrac{1}{2}(AT/q)]^{-1}[I_n + \tfrac{1}{2}(AT/q)]\}^q \triangleq G_{3,q} \tag{5c}$$

$$\simeq \cdots$$

In general, the approximants of $\exp(AT)$ can be written as

$$G_{i,q} = \begin{cases} \left\{\left[\sum_{j=0}^{i/2-1} \frac{(i-1-j)!(i/2-1)!}{(i-1)!j!(i/2-1-j)!}\left(-\frac{AT}{q}\right)^j\right]^{-1}\left[\sum_{j=0}^{i/2} \frac{(i-1-j)!(i/2)!}{(i-1)!j!(i/2-1-j)!}\left(\frac{AT}{q}\right)^j\right]\right\}^q \\ \text{for } i = 2, 4, 6, \ldots \end{cases} \tag{5d}$$

$$\begin{cases} \left\{\left[\sum_{j=0}^{(i-1)/2} \frac{(i-1-j)!((i-1)/2)!}{(i-1)!j!((i-1)/2-j)!}\left(-\frac{AT}{q}\right)^j\right]^{-1}\left[\sum_{j=0}^{(i-1)/2} \frac{(i-1-j)!((i-1)/2)!}{(i-1)!j!((i-1)/2-j)!}\left(\frac{AT}{q}\right)^j\right]\right\}^q \\ \text{for } i = 3, 5, 7, \ldots \end{cases} \tag{5e}$$

The first subscript i of $G_{i,q}$ in equation (5) denotes the number of the quotients in the continued fraction expansion of $\exp(AT/q)$ to be taken and the second subscript q of $G_{i,q}$ is the scaling and squaring factor which is often chosen as a power of two. For the analysis of the truncation error and the determination of the q, we define a relative error matrix and compute its approximant as

$$E_G \triangleq [G_{i,q} - G]G^{-1} \simeq q\left(\frac{AT}{q}\right)^i \Big/ K, \tag{6a}$$

where

$$K = \begin{cases} \dfrac{(-1)^{i/2} i!(i-1)!}{(i/2)!(i/2-1)!}, & \text{for } i = 2, 4, 6, \ldots, \end{cases} \tag{6b}$$

$$\begin{cases} \dfrac{(-1)^{(i+1)/2} i!(i-1)!}{((i-1)/2)!((i-1)/2)!}, & \text{for } i = 3, 5, 7, \ldots. \end{cases} \tag{6c}$$

With the prescribed values such as the sampling period T, the value i in equations (5) and (6) and the relative error tolerance $||E_G||$ in equation (6a), we can determine the scaling and squaring factor q in equations (5) from (6) as

$$q = (||(AT)^i||/(|K|\,||E_G||))^{1/(i-1)}. \tag{7}$$

Note that a small $||E_G||$ results in a large q. When the system matrix A in equations (5) is a stiff matrix and the power j is a large value, the computation of the high order approximant in equations (5) often produces round-off errors due to the computation of $(AT/q)^j$. To avoid the above numerical difficulty, we shall develop a stable and fast recursive algorithm for computing the matrix G from the matrix A in Section 4.

(b) The square-root, scaling, continued-fraction method for finding a continuous-time model from a discrete-time model [12]

The equivalent continuous-time model in equation (1) can be obtained from the discrete-time model in equations (3) as follows:

$$A = \frac{1}{T}\ln(G) \tag{8a}$$

and

$$B = A[G - I_n]^{-1}H. \tag{8b}$$

The equivalent continuous-time system matrix A in equation (8a) can be obtained by truncating the following infinite series as

$$A = \frac{1}{T}\ln(G) = \frac{2}{T}[R + \tfrac{1}{3}R^3 + \tfrac{1}{5}R^5 + \cdots] = \frac{2}{T}\sum_{j=0}^{\infty}\frac{1}{2j+1}R^{2j+1}, \tag{9}$$

where $R = [G - I_n][G + I_n]^{-1}$ with $\mathrm{Re}[\sigma(G)] > 0$, where $\sigma(G)$ is the spectrum of the matrix G.

When $\mathrm{Re}[\sigma(G)] \to +1$ and $\mathrm{Im}[\sigma(G)] \to 0$, then $|\sigma(R)| \ll 1$. Hence, the truncation of the above series will give a good approximant. Since not all eigenvalues of the matrix G always lie in the right half complex plane (i.e. $\mathrm{Re}[\sigma(G)] > 0$) and close to a positive unity; therefore, we need to modify the matrix G before we carry out the truncation of the infinite series in equation (9). Taking the qth root of the matrix G in equation (5a) results in

$$G^{1/q} = [\exp(AT)]^{1/q} = \exp(AT/q). \tag{10a}$$

Thus,

$$A = q\left[\frac{1}{T}\ln(G^{1/q})\right] \text{ with } q \text{ as a power of two.} \tag{10b}$$

If the matrix $G^{1/q}$ is the *principal* qth root [15, 16] of the non-singular matrix G, where the *principal* qth root of a matrix G has the properties that $(G^{1/q})^q = G$ and $\arg(\sigma(G^{1/q})) \epsilon (-\pi/q, +\pi/q)$ for $q \geqslant 2$, then the matrix A in equation (10b) can be approximated by truncating the following infinite series as follows:

$$A = \frac{1}{T}\ln(G) = \frac{q}{T}\ln(G^{1/q}), \tag{11a}$$

where

$$\ln(G^{1/q}) = 2[\hat{R} + \tfrac{1}{3}\hat{R}^3 + \tfrac{1}{5}\hat{R}^5 + \cdots] = 2\sum_{j=0}^{\infty}\frac{1}{2j+1}\hat{R}^{2j+1}, \tag{11b}$$

$$\simeq 2\hat{R} \simeq 2[\hat{R} + \tfrac{1}{3}\hat{R}^3] \simeq \cdots \tag{11c}$$

and

$$\hat{R} \triangleq [G^{1/q} - I_n][G^{1/q} + I_n]^{-1}. \tag{11d}$$

For a large q, $\mathrm{Re}[\sigma(G^{1/q})] \to +1$ and $\mathrm{Im}[\sigma(G^{1/q})] \to 0$. Since q is chosen as the power of two, the principal qth root of the matrix G can be found by successively taking its principal square root of the matrix G. A numerically *stable* recursive algorithm [17, 18] with a quadratic convergence rate can be applied to determine $G^{1/q}$ and is summarized as follows:

$$P(k+1) = \tfrac{1}{2}[P(k) + Q(k)^{-1}], \quad P(0) = G \tag{12a}$$

$$Q(k+1) = \tfrac{1}{2}[P(k)^{-1} + Q(k)], \quad Q(0) = I_n \tag{12b}$$

$$\lim_{k\to\infty} P(k) = G^{1/2} \tag{12c}$$

$$\lim_{k\to\infty} Q(k) = (G^{1/2})^{-1}. \tag{12d}$$

The algorithm in equations (12) fails [16, 17, 18] when the matrix G contains any negative real eigenvalues. In this case the matrix G can be rotated by a small positive real angle $\Delta\theta$ to give $\hat{G} = G\exp(-j\Delta\theta)$ so that $G^{1/2} = \hat{G}^{1/2}\exp(j\Delta\theta/2)$.

From equation (10a) we observe that when the continuous-time system matrix AT is normalized by a *scaling* factor q, the equivalent discrete-time matrix becomes $G^{1/q}$. Therefore, the *scaling* factor q for the continuous-time system matrix becomes the *square-root* factor for the discrete-time system

matrix. Also, from equation (10b) it can be seen that the *scaling* factor q for the discrete-time system matrix becomes the *squaring* factor for the continuous-time system matrix in equation (5a). Once we have the matrix $G^{1/q}$, we can determine the approximant of $\ln(G^{1/q})$ via the continued-fraction method [12]. Some approximants of $\ln(G^{1/q})$ obtained by the square-root, scaling, continued-fraction method can be written as follows:

$$A = \frac{1}{T}\ln(G) = \frac{q}{T}\ln(G^{1/q}), \tag{13a}$$

where

$$\frac{1}{T}\ln(G^{1/q}) \simeq \frac{1}{T}2\hat{R} \triangleq A_{1,q} \tag{13b}$$

$$\simeq \frac{1}{T}2\hat{R}[I_n - \tfrac{1}{3}\hat{R}^2]^{-1} \triangleq A_{2,q} \tag{13c}$$

$$\simeq \frac{1}{T}2\hat{R}[I_n - \tfrac{4}{15}\hat{R}^2][I_n - \tfrac{3}{5}\hat{R}^2]^{-1} \triangleq A_{3,q} \tag{13d}$$

$$\simeq \cdots,$$

where the matrix $\hat{R}$ is defined in expression (11d).

The first subscript i of $A_{i,q}$ denotes that i quotients of the continued-fraction expansion of $\ln(G^{1/q})$ have been used for finding the approximant of $\ln(G^{1/q})$, whereas the second subscript q of $A_{i,q}$ is the square-root and scaling factor. For the analysis of the truncation error and the determination of the q, we define a relative error matrix and compute its approximant as

$$E_A \triangleq [qA_{i,q} - A]A^{-1} \simeq -L\hat{R}^{2i} \simeq L[G + G^{-1} - 2I_n]^i/(2q)^{2i}, \tag{14a}$$

where

$$L = \begin{cases} \left[\prod_{j=0}^{(i-1)/2} 2^{2j}\right][(i+2)^2(i+4)^2 \ldots (2i-1)^2(2i+1)]^{-1}, & \text{for } i = 1, 3, 5, \ldots, \\ \left[\prod_{j=0}^{i/2} 2^{2j}\right][(i+1)^2(i+3)^2 \ldots (2i-1)^2(2i+1)]^{-1}, & \text{for } i = 2, 4, 6, \ldots. \end{cases} \tag{14b}$$

Having prescribed the values for the sampling period T, the value of i in equation (13) and the relative error tolerance $||E_A||$ in equations (14), the square-root and scaling factor q can be determined as follows:

$$q = \tfrac{1}{2}(L||(G + G^{-1} - 2I_n)^i||/||E_A||)^{1/2i}. \tag{15}$$

A small $||E_A||$ results in a large q. When the matrix G is a stiff matrix, the computation of $G^{1/q}$ with a high power q in equations (12) may induce round-off errors. To overcome the above difficulties, the development of a fast and stable recursive algorithm for finding the matrix A from the matrix G becomes necessary and is shown as follows.

3. FAST AND STABLE RECURSIVE ALGORITHM FOR FINDING A FROM G

The recursive algorithm developed in Refs [1, 2] for finding the matrix A from the matrix G can be reviewed as follows.

The desired matrix A can be solved from the following matrix equation:

$$F_1(A) = \exp(-AT) - G^{-1} = 0_n \tag{16a}$$

where the matrix 0_n is an $n \times n$ null matrix. Following the Newton–Raphson method yields

$$A(k+1) = A(k) - [F'(A(k))]^{-1}F(A(k)). \tag{16b}$$

Substituting the Jacobian matrix, $F'(A(k)) = -T\exp(-A(k)T)$ which is a non-linear matrix-

valued function, in equation (16b) gives

$$A(k+1) = A(k) + \frac{1}{T}[I_n - \exp(A(k)T)G^{-1}], \quad \text{for } k = 0, 1, 2, \ldots, \tag{16c}$$

$$A(0) = \frac{1}{T} 2(G - I_n)(G + I_n)^{-1} \tag{16d}$$

$$\lim_{k \to \infty} A(k) = A. \tag{16e}$$

At each iteration, the approximant of $\exp(A(k)T)$ via the direct truncation method in equation (4) is determined and substituted into equation (16c) for finding the up-dated $A(k+1)$. In Refs [1, 2, 11], it was stated that the desired matrix A can be determined when $|\sigma(A(k))T| \leqslant 0.5$ and the algorithm in equations (16) may become *unstable* for a given matrix G if the sampling period T is too large. The inverse of the matrix G in equation (16c) can be avoided if we use an alternative matrix equation given as

$$F_2(A) = -\exp(AT) + G = 0_n. \tag{17a}$$

Thus, the corresponding recursive algorithm becomes

$$A(k+1) = A(k) - \frac{1}{T}[I_n - \exp(-A(k)T)G], \quad \text{for } k = 0, 1, 2, \ldots, \tag{17b}$$

$$A(0) = \frac{2}{T}(G - I_n)(G + I_n)^{-1} \tag{17c}$$

$$\lim_{k \to \infty} A(k) = A. \tag{17d}$$

The accuracy of the Newton–Raphson algorithms shown in equations (16) and (17) is heavily dependent on the approximant of $\exp(A(k)T)$ or $\exp(-A(k)T)$. In this section, we shall develop a rapidly convergent and numerically stable recursive algorithm for the model conversion of the system which may not satisfy the conditions $\mathrm{Re}[\sigma(G)] > 0$ and/or $|\sigma(A)T| \leqslant 0.5$ with a *fixed* sampling period T.

It is well known that the convergence of the Newton–Raphson algorithm depends heavily upon the initial guess of the recursive algorithm, and the property of the Jacobian matrix (the slope) of a matrix-valued function (a scalar-valued function). For example, we consider two scalar-valued non-linear functions $f_1(a) = \exp(-aT) - g^{-1}$ and $f_2(a) = -\exp(aT) + g$ where T and g are constants and a is a variable. The functions $f_1(a)$ and $f_2(a)$ have the slopes $-T/g$ and $-Tg$, respectively, at the solution $a = 1/T \ln(g)$. In this paper, we propose a new function $h(a) = \ln(\exp(-aT)g) = -aT + \ln(g)$ which is a linear function with a constant slope $-T$ for any a including the solution $a = 1/T \ln(g)$. To compare the convergence speeds of the Newton–Raphson algorithms developed for the non-linear functions $f_1(a)$ and $f_2(a)$ with the convergence speed for the linear function $h(a)$, we modify the functions $f_1(a)$ and $f_2(a)$ as

$$\hat{f}_1(a) = (\exp(-aT) - g^{-1})g = \exp(-aT)g - 1,$$

and

$$\hat{f}_2(a) = (-\exp(aT) + g)g^{-1} = -\exp(aT)g^{-1} + 1,$$

respectively, so that the modified non-linear functions $\hat{f}_1(a)$ and $\hat{f}_2(a)$ have the same slope, $-T$, as that of $h(a)$ at the solution $a = 1/T \ln(g)$. The functions $\hat{f}_1(a)$ and $\hat{f}_2(a)$ are indeed the approximants of the newly proposed function $h(a)$ as follows:

$$\begin{aligned} h(a) &= \ln(\exp(-aT)g) = \ln[(\exp(-aT)g - 1) + 1] \\ &= [\exp(-aT)g - 1] - \tfrac{1}{2}[\exp(-aT)g - 1]^2 \\ &\quad + \tfrac{1}{3}[\exp(-aT)g - 1]^3 - \cdots \simeq \exp(-aT)g - 1 = \hat{f}_1(a). \end{aligned} \tag{18a}$$

Also, in the same fashion, we obtain

$$h(a) = \ln[\exp(-aT)g] = -\ln[\exp(aT)g^{-1}] \simeq -\exp(aT)g^{-1} + 1 = \hat{f}_2(a). \tag{18b}$$

Another approximant of the function $h(a)$, defined as $h_1(a)$, can be obtained as follows:

$$\begin{aligned} h(a) = \ln[\exp(-aT)g] &= 2\{[(\exp(-aT)g - 1)(\exp(-aT)g + 1)^{-1}] \\ &\quad + \tfrac{1}{3}[(\exp(-aT)g - 1)(\exp(-aT)g + 1)]^3 + \cdots\} \\ &\simeq 2(\exp(-aT)g - 1)(\exp(-aT)g + 1)^{-1} \triangleq h_1(a). \end{aligned} \tag{18c}$$

In addition, an alternative approximant of the function $h(a)$, defined as $h_2(a)$, can be written as

$$\begin{aligned} h(a) &= \ln[\exp(-aT)g] \\ &= \ln[(\exp(-aT)g)^{1/2}/(\exp(-aT)g)^{-1/2}] \simeq (\exp(-aT)g)^{1/2} \\ &\quad - (\exp(-aT)g)^{-1/2} \triangleq h_2(a). \end{aligned} \tag{18d}$$

Note that all functions in equations (18) have the same solution at $a = 1/T \ln(g)$ and their respective slopes for any a are

$$\begin{aligned} \hat{f}_1'(a) &= -T \exp(-aT)g, \\ \hat{f}_2'(a) &= -T \exp(aT)g^{-1}, \\ h'(a) &= -T, \\ h_1'(a) &= -4T \exp(-aT)g(\exp(-aT)g + 1)^{-2} \end{aligned}$$

and

$$h_2'(a) = -T[(\exp(-aT)g)^{1/2} + (\exp(-aT)^{-1/2}]/2.$$

The curves of $\hat{f}_1(a)$, $\hat{f}_2(a)$, $h(a)$, $h_1(a)$ and $h_2(a)$ are shown in Fig. 1. From Fig. 1, we observe that the curve $h(a)$ is a straight line and the curves $h_1(a)$ and $h_2(a)$ are closer to the curve $h(a)$ than those of $\hat{f}_1(a)$ and $\hat{f}_2(a)$. Thus, if we can determine the exact values of $h(a)$, $h_i(a)$ and $\hat{f}_i(a)$ for

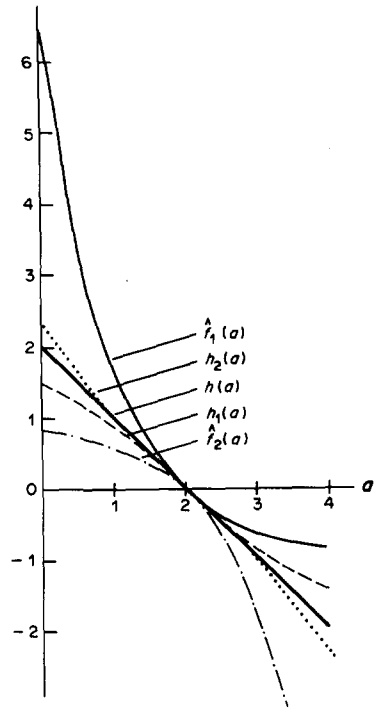

Fig. 1. Plots of various functions.

$i = 1, 2$ at an arbitrary initial guess $a = a_0$; then, we shall obtain the solution $a = 1/T \ln(g)$ at one step for the linear function $h(a)$, a few steps for the functions $h_i(a)$ and many steps for the functions $\hat{f}_i(a)$ with $i = 1, 2$. Thus, the convergence speeds of the Newton–Raphson algorithms developed for the functions $h_i(a)$ are faster than those of the functions $\hat{f}_i(a)$ for $i = 1, 2$. In other words, the convergence speed of the Newton–Raphson algorithm depends on the approximation of the logarithmic function, $\ln[\exp(-aT)g]$, and the accuracy of the recursive algorithm depends on the approximant of the exponential function, $\exp(-aT)$.

Based on the above reasoning, we propose a new matrix-valued function $H(A)$ to improve the recursive algorithms shown in equations (16) and (17) as follows:

$$H(A) = \ln[\exp(AT)G^{-1}] = -\ln[\exp(-AT)G] = -\ln[I_n] = 0_n. \tag{19a}$$

The corresponding Newton–Raphson algorithm becomes

$$A(k+1) = A(k) - [H'(A)]^{-1}H(A)|_{A=A(k)} \tag{19b}$$

$$= A(k) - \frac{1}{T}\ln[\exp(A(k)T)G^{-1}], \quad \text{for } k = 0, 1, 2, \ldots, \tag{19c}$$

$$= A(k) + \frac{1}{T}\ln[\exp(-A(k)T)G], \quad \text{for } k = 0, 1, 2, \ldots, \tag{19d}$$

$$A(0) = 0_n \tag{19e}$$

$$\lim_{k\to\infty} A(k) = A. \tag{19f}$$

Note that the first iteration yields $A(1) = 1/T \ln(G)$. Because the Jacobian matrix, $H'(A(k)) = TI_n$, is a constant diagonal matrix, the convergence speed of the recursive algorithm in equation (19d) depends on the approximant of $\ln[\exp(-A(k)T)G]$, whereas the accuracy of the recursive algorithm depends on the approximant of $\exp(-A(k)T)$. Since

$$\ln[\exp(-A(k)T)G] \simeq I_n - \exp(A(k)T)G^{-1},$$

the recursive algorithms in equations (16) and (17) are a special case of the algorithms in equations (19d) and (19c), respectively.

The logarithmic matrix-valued functions in equations (19c) and (19d) can be approximated via either the direct truncation method in equation (11) or the square-root, scaling and continued-fraction method in equation (13) with $q \geqslant 1$. For example, when $q = 2$ and the simplest approximant in equations (11) or (13) is used, we obtain

$$\hat{A}(k+1) = \hat{A}(k) + \frac{2}{T}[\exp(-\hat{A}(k)T)\hat{G} - I_n][\exp(-\hat{A}(k)T)\hat{G} + I_n]^{-1}, \quad \hat{A}(0) = 0_n \tag{20}$$

where $\hat{A}(k) \triangleq \mathrm{A(k)}/2$ and $\hat{G} \triangleq \mathrm{G}^{1/2}$. The $\hat{G}$ can be determined via the stable algorithm in equations (12). The direct truncation method in equation (4) or the scaling, squaring and continued-fraction method in equation (5) can be applied to determine the approximant of $\exp(-\hat{A}(k)T)$. Note that for the algorithm in equation (20), the first iteration gives $\hat{A}(1) = 2/T[\hat{G} - I_n][\hat{G} + I_n]^{-1}$, which is the initial condition used in the algorithms in equations (16) and (17). It can be shown via the practical example in Section 5 that both algorithms in equations (16) and (20) are numerically unstable. To overcome the instability, a new recursive algorithm is developed for finding A from G as follows:

Define $M(k) \triangleq \exp(-\hat{A}(k)T)\hat{G}$. From equation (19d) we can derive the equations as follows:

$$\exp(-\hat{A}(k+1)T)\hat{G} = \exp(-\hat{A}(k)T)\hat{G}\exp(-\ln(\exp(-\hat{A}(k)T)G)),$$

i.e.

$$M(k+1) = M(k)\exp(-\ln(M(k))), \quad M(0) = G^{1/q} \tag{21a}$$

and

$$\hat{A}(k+1) = \hat{A}(k) + \frac{1}{T}\ln(M(k)), \quad \hat{A}(0) = 0_n \tag{21b}$$

$$\lim_{k\to\infty} M(k) = I_n \tag{21c}$$

$$\lim_{k\to\infty} q\hat{A}(k) = A. \tag{21d}$$

The following approximants can be used to approximate $\ln(M(k))$ in equation (21):

$$\ln(M(k)) \simeq M(k) - I_n \tag{22a}$$

$$\simeq [M(k) - I_n] - \tfrac{1}{2}[M(k) - I_n]^2 \tag{22b}$$

$$\simeq \cdots,$$

or

$$\ln(M(k)) \simeq 2R(k) \tag{22c}$$

$$\simeq 2R(k)[I_n - \tfrac{4}{15}R(k)^2][I_n - \tfrac{3}{5}R(k)^2]^{-1} \tag{22d}$$

$$\simeq \cdots,$$

where

$$R(k) = [M(k) - I_n][M(k) + I_n]^{-1}.$$

If the approximant in expression (22a) is used, the algorithm in equations (21) becomes

$$M(k+1) = M(k)\exp[I_n - M(k)], \quad M(0) = G^{1/q} \tag{23a}$$

$$\hat{A}(k+1) = \hat{A}(k) - \frac{1}{T}[I_n - M(k)], \quad \hat{A}(0) = 0_n \tag{23b}$$

$$\lim_{k\to\infty} M(k) = I_n \tag{23c}$$

$$\lim_{k\to\infty} q\hat{A}(k) = A. \tag{23d}$$

If the approximant in expression (22b) is used, the algorithm in equations (21) becomes

$$M(k+1) = M(k)\exp[(I_n - M(k)) + \tfrac{1}{2}(I_n - M(k))^2], \quad M(0) = G^{1/q} \tag{24a}$$

$$\hat{A}(k+1) = \hat{A}(k) - \frac{1}{T}[(I_n - M(k)) + \tfrac{1}{2}(I_n - M(k))^2], \quad \hat{A}(0) = 0_n \tag{24b}$$

$$\lim_{k\to\infty} M(k) = I_n \tag{24c}$$

$$\lim_{k\to\infty} q\hat{A}(k) = A. \tag{24d}$$

If the approximant in equation (22c) is utilized, the algorithm in equations (21) becomes

$$M(k+1) = M(k)\exp[-2R(k)], \quad M(0) = G^{1/q} \tag{25a}$$

$$\hat{A}(k+1) = \hat{A}(k) + \frac{2}{T}R(k), \quad \hat{A}(0) = 0_n \tag{25b}$$

$$\lim_{k\to\infty} M(k) = I_n \tag{25c}$$

$$\lim_{k\to\infty} q\hat{A}(k) = A. \tag{25d}$$

Also, if the approximant in expression (22d) is employed, the algorithm in equations (21) gives

$$M(k+1) = M(k)\exp\{-2R(k)[I_n - \tfrac{4}{15}R(k)^2][I_n - \tfrac{3}{5}R(k)^2]^{-1}\}, \quad M(0) = G^{1/q} \tag{26a}$$

$$\hat{A}(k+1) = \hat{A}(k) + \frac{2}{T}R(k)[I_n - \tfrac{4}{15}R(k)^2][I_n - \tfrac{3}{5}R(k)^2]^{-1}, \quad \hat{A}(0) = 0_n \tag{26b}$$

$$\lim_{k\to\infty} M(k) = I_n \tag{26c}$$

$$\lim_{k\to\infty} q\hat{A}(k) = A. \tag{26d}$$

The matrix exponential functions in equations (23a), (24a), (25a) and (26a) can be computed via the techniques shown in Section 2, while the initial conditions $G^{1/q}$ can be obtained using the algorithm in equations (12). The algorithms in equations (23), (24), (25) and (26) are numerically stable and the stability is proved in Appendix A. Note that the accuracy of the recursive algorithm in equations (21) depends upon the approximation of the matrix exponential function, while the convergence speed of the recursive algorithm in equations (21) depends on the approximation of the matrix logarithmic function and the factor q.

Substituting the required system matrix A and the available matrix G to equation (8b) yields the input matrix B. Thus, we obtain the desired continuous-time model in equation (1).

4. FAST AND STABLE RECURSIVE ALGORITHM FOR FINDING G FROM A

From the newly developed function in equation (19a) for finding A from G, we can construct the following matrix equation for finding G from A:

$$E(G) = \exp\{\ln[\exp(-AT)G]\} - \exp[\ln(I_n)] = \exp[\ln(G) - AT] - I_n = 0_n. \tag{27a}$$

The corresponding Newton–Raphson algorithm becomes

$$G(k+1) = G(k) - [E'(G)]^{-1}E(G)|_{G=G(k)} \tag{27b}$$

$$= G(k)\exp[AT - \ln(G(k))], \quad \text{for } k = 0, 1, 2, \ldots, \tag{27c}$$

$$G(0) = I_n \tag{27d}$$

and

$$\lim_{k\to\infty} G(k) = G. \tag{27e}$$

Note that the first iteration gives $G(1) = \exp(AT)$. The Jacobian matrix,

$$E'(G) = \{\exp[\ln(G) - AT]\}G^{-1} = \exp(-AT)$$

is a constant matrix. Therefore, the convergence speed depends upon the approximant of $\exp[-AT + \ln(G(k))]$ and the accuracy of the recursive algorithm depends on the approximant of $\ln(G(k))$. Because the series expansion of $\ln(G)$ in equation (9) for $\mathrm{Re}[\sigma(G)] > 0$ converges very fast, good approximants of $\ln(G(k))$ can be determined even if the matrix G is a stiff matrix. As a result, $\sigma(AT - \ln(G(k))] \simeq 0$ and good approximants of $\exp[AT - \ln(G(k))]$ can be obtained via the methods shown in equation (4) and (5). The imaginary parts of the eigenvalues of AT may be such that $\pi > |\mathrm{Im}(\sigma(AT))| \geqslant \pi/2$. As a result, $\mathrm{Re}(\sigma(G))$ may be negative and the infinite series expansion of $\ln(G)$ in equation (9) will not converge. To overcome this difficulty, we normalize the matrix A by a scaling factor $q \geqslant 2$ so that $\pi/2 > |\mathrm{Im}(\sigma(AT))| \geqslant 0$ and $\mathrm{Re}(\sigma(\hat{G})) > 0$, where $\hat{A} \triangleq A/q$ and $\hat{G} \triangleq \exp(\hat{A}T)$. Thus, the infinite series expansion of $\ln(\hat{G})$ in equation (11b) converges. The Newton–Raphson algorithm in equation (27c) becomes

$$\hat{G}(k+1) = \hat{G}(k)\exp[\hat{A}T - \ln(\hat{G}(k))], \quad \text{for } k = 0, 1, 2, \ldots, \tag{28a}$$

$$\hat{G}(0) = I_n \tag{28b}$$

and

$$\lim_{k\to\infty} (\hat{G}(k))^q = G. \tag{28c}$$

To derive the numerically stable algorithm we modify equation (28a) as

$$\hat{A}T - \ln[\hat{G}(k+1)] = \hat{A}T - \ln[\hat{G}(k)] - \ln\{\exp[\hat{A}T - \ln(\hat{G}(k))]\}. \tag{29}$$

Defining

$$N(k) \triangleq \hat{A}T - \ln(\hat{G}(k)) \tag{30}$$

and substituting expression (30) into equation (29) and equations (28) yields

$$N(k+1) = N(k) - \ln[\exp(N(k))], \quad N(0) = AT/q \tag{31a}$$

$$\hat{G}(k+1) = \hat{G}(k)\exp(N(k)), \quad \hat{G}(0) = I_n \tag{31b}$$

$$\lim_{k\to\infty} N(k) = 0_n \tag{31c}$$

$$\lim_{k\to\infty} (\hat{G}(k))^q = \exp(AT) = G. \tag{31d}$$

Since

$$\exp(X) \simeq I_n + X \simeq [I_n + \tfrac{1}{2}X][I_n - \tfrac{1}{2}X]^{-1} \simeq \cdots, \tag{32}$$

where X is a matrix. The algorithm in equations (31) can be represented by various forms as follows:

$$N(k+1) = N(k) - \ln[I_n + N(k)], \quad N(0) = AT/q \tag{33a}$$

$$\hat{G}(k+1) = \hat{G}(k)[I_n + N(k)], \quad \hat{G}(0) = I_n \tag{33b}$$

$$\lim_{k\to\infty} N(k) = 0_n \tag{33c}$$

$$\lim_{k\to\infty} (\hat{G}(k))^q = \exp(AT) = G. \tag{33d}$$

Also,

$$N(k+1) = N(k) - \ln\{[I_n + \tfrac{1}{2}N(k)][I_n - \tfrac{1}{2}N(k)]^{-1}\}, \quad N(0) = AT/q \tag{34a}$$

$$\hat{G}(k+1) = \hat{G}(k)[I_n + \tfrac{1}{2}N(k)][I_n - \tfrac{1}{2}N(k)]^{-1}, \quad \hat{G}(0) = I_n \tag{34b}$$

$$\lim_{k\to\infty} N(k) = 0_n \tag{34c}$$

$$\lim_{k\to\infty} (\hat{G}(k))^q = \exp(AT) = G \tag{34d}$$

and

$$N(k+1) = N(k) - \ln\{[I_n + \tfrac{1}{2}N(k) + \tfrac{1}{12}N(k)^2][I_n - \tfrac{1}{2}N(k) + \tfrac{1}{12}N(k)^2]^{-1}\}, \quad N(0) = AT/q \tag{35a}$$

$$\hat{G}(k+1) = \hat{G}(k)[I_n + \tfrac{1}{2}N(k) + \tfrac{1}{12}N(k)^2][I_n - \tfrac{1}{2}N(k) + \tfrac{1}{12}N(k)^2]^{-1}, \quad \hat{G}(0) = I_n \tag{35b}$$

$$\lim_{k\to\infty} N(k) = 0_n \tag{35c}$$

$$\lim_{k\to\infty} (\hat{G}(k))^q = \exp(AT) = G. \tag{35d}$$

The matrix logarithmic functions in equations (33a), (34a) and (35a) can be computed using the techniques shown in Section 2. The algorithms in equations (33), (34) and (35) are numerically stable. The stability is proved in Appendix B.

Substituting the obtained matrix G and the available matrix A to equation (3c) gives the input matrix H. Thus, we obtain the discrete-time model in equation (3a).

5. ILLUSTRATIVE EXAMPLE

A practical example, which consists of a stiff system matrix, is presented for finding the matrix $A(G)$ from the matrix $G(A)$ as follows.

Consider the stiff continuous-time system matrix of a linearized two shaft gas-turbine model [19]:

$$A = \begin{bmatrix} -1.268 & -0.04528 & 1.498 & 951.5 \\ 1.002 & -1.957 & 8.52 & 1240 \\ 0 & 0 & -10 & 0 \\ 0 & 0 & 0 & -100 \end{bmatrix},$$

where $\sigma(A) = \{-1.3417, -1.8833, -10, -100\}$ and $\sigma(A)T = \{-0.05367, -0.07533, -0.4, -4\}$

having a fixed sampling period $T = 0.04$ s. Note that $|\sigma(AT)| > 0.5$. The associated discrete-time system matrix with the same fixed sampling period $T = 0.04$ s is

$$G = \begin{bmatrix} 0.9505105993 & -0.0016980986 & 0.0478132051 & 8.967804795 \\ 0.0375771817 & -0.9246715991 & 0.2704783066 & 11.7361807569 \\ 0 & 0 & 0.6703200460 & 0 \\ 0 & 0 & 0 & 0.0183156389 \end{bmatrix},$$

where $\sigma(G) = \{0.94774510, 0.92743710, 0.67032005, 0.01831564\}$.

It is desired to find the matrix A from the matrix G via the algorithms in equations (16), (20), (23), (24) and (25). Since $\mathrm{Re}[\sigma(G)] > 0$, the square-root factor q can be chosen as 1. The infinite series approximation method shown in equation (4) with $j = 31$ is employed for finding the approximants of the matrix exponential functions in equations (16), (20), (23), (24) and (25). The simulation results are shown in Fig. 2. The algorithms in equations (16) and (20) converge at the 9th iteration with $||(A(9) - A)A^{-1}|| = 3.4 \times 10^{-5}$ and at the 5th iteration with $||(A(5) - A)A^{-1}|| = 7 \times 10^{-11}$, respectively, then both algorithms diverge. Therefore, the algorithms in equations (16) and (20) are numerically unstable. The algorithms in equations (23), (24) and (25) converge at the 9th iteration with $||(A(9) - A)A^{-1}|| = 4.4 \times 10^{-15}$, at the 6th iteration with $||(A(6) - A)A^{-1}|| = 2.2 \times 10^{-15}$ and at the 5th iteration with $||(A(5) - A)A^{-1}|| = 3.6 \times 10^{-14}$, respectively. All three algorithms are numerically stable. Since the computational loads of the proposed rapidly convergent algorithms in equations (24) and (25) are greater than that of the existing algorithms in equations (16), it might be interesting to compare the CPU times of the algorithms in equations (23), (24), (25) and (16). With the same relative error level (1×10^{-5}), the CPU times for the algorithm in equations (23) with 7 iterations, the algorithm in equations (24) with 5 iterations, the algorithm in equations (25) with 4 iterations and the algorithm in equations (16) with 9 iterations using the VAX 11/750 computer are 0.007, 0.005, 0.28 and 0.14 ms, respectively. It is observed that the proposed algorithms in equations (23) and (24) give faster computation time than the existing algorithm in equations (16) which involves two matrix inversions for computing the initial condition and G^{-1}, while the proposed algorithm in equations (25) provides slower computation time than the algorithms in equations (23), (24) and (16) because it involves four matrix inversions (i.e. one matrix inversion for each iteration). From above results we conclude that a rapidly convergent algorithm may not provide faster computation time due to computational complexity of the algorithm. However, the computational time due to computational complexity can be improved by using advanced computer technologies such as the parallel computational techniques [20], etc.

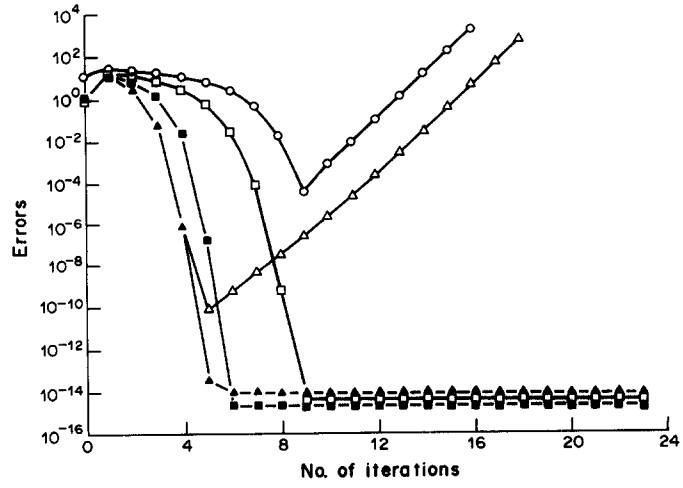

Fig. 2. Finding A from G: ○—algorithm in equations (16); △—algorithm in equation (20); □—algorithm in equations (23); ■—algorithms in equations (24); ▲—algorithms in equations (25).

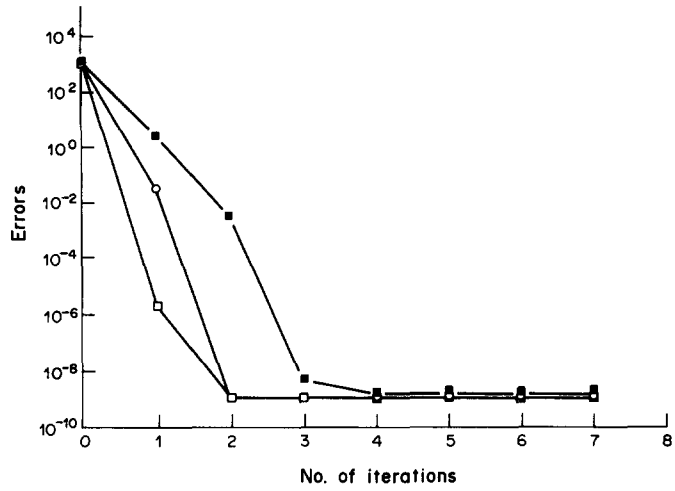

Fig. 3. Finding G from A: ■—algorithm in equations (33); ○—algorithm in equations (34); □—algorithm in equations (35).

To satisfy the condition $|\sigma(AT)| \leqslant 0.5$ in the algorithm in equation (16), we select $T = 0.005$ s and carry out the simulations using algorithms (16), (23) and (24). The algorithms in equations (16), (23) and (24) converge at the 5th iteration with a relative error 2.1×10^{-14}, at the 5th iteration with a relative error 1.3×10^{-14} and at the 3rd iteration with a relative error 6.4×10^{-14}, respectively. The corresponding computation times are 0.14, 0.005 and 0.003 ms, respectively. All three algorithms are stable; however, the proposed algorithms in equations (23) and (24) provide faster computation times than the algorithm in equations (16) which involves two matrix inversions.

To find the matrix G from the matrix A we use the stable algorithms in equations (33), (34) and (35) as follows.

Let $\ln(X) \simeq 2R[I_n - \frac{4}{15}R^2][I_n - \frac{3}{15}R^2]^{-1}$, where $R = [X - I_n][X + I_n]^{-1}$ and $q = 64$. With the same relative error tolerance $||(G(k) - G)G^{-1}|| \simeq 1 \times 10^{-9}$, the algorithm in equations (33) converges at $k = 4$, while the algorithms in equations (34) and (35) converge at $k = 2$. The simulation results are shown in Fig. 3. For the purpose of comparison, the scaling, squaring continued-fraction method in equations (5), which is a non-recursive algorithm, has been employed to compute the approximant $G_{i,q}$ with $i = 5$ and the *same* value of $q(=64)$. The associated relative error is 2×10^{-4}. The corresponding CPU times for the algorithms in equations (33), (34) and (5) for the *same* $q(=64)$ are 0.56, 0.42 and 0.1 ms, respectively, while the corresponding relative errors for the algorithms are 1×10^{-9}, 1×10^{-9} and 2×10^{-4}, respectively. The non-recursive algorithm in equations (5) provides faster computation time than the proposed recursive algorithms which involve matrix inversions; however, the proposed recursive algorithms give smaller errors than the non-recursive algorithm for the same q. The use of a small q will result in a small round-off error caused by computing the matrix $(G_q)^q$, where the matrix G_q is an approximant of the $\exp(AT/q)$ in equation (5a) or the matrix $(\hat{G}_q(k))^q$ where the matrix

$$\hat{G}_q(k) = \lim_{k \to \infty} (\hat{G}(k))$$

in equation (28c).

6. CONCLUSION

Numerically stable and rapidly convergent recursive algorithms for modelling the equivalent continuous-time (discrete-time) model from the available discrete-time (continuous-time) model have been presented. The non-recursive algorithms such as the scaling, squaring continued-fraction method and the square-root, scaling, continued-fraction method for the models conversion have been introduced, and their truncation errors have been analyzed. Based on the Newton–Raphson method, new recursive algorithms have been developed for continuous-time and discrete-time

model conversions. It is shown that the proposed algorithms are numerically stable and that the convergence speed of the recursive algorithm for obtaining the continuous-time (discrete-time) system matrix from the discrete-time (continuous-time) system matrix depends upon the approximation of the matrix logarithmic (exponential) function, whereas the accuracy of the recursive algorithm depends on the approximation of the matrix exponential (logarithmic) function. The newly developed recursive algorithms relax the constraints imposed on the existing model conversion methods and, hence, can be applied to the system which may contain both large and small eigenvalues with a fixed sampling period. A practical example has been presented to demonstrate the effectiveness of the proposed procedures. Thus, utilizing the proposed model conversion techniques, we can analyze and design a sampled-data system via the well-developed continuous-time or discrete-time techniques [2, 21].

It should be noted that the proposed algorithms do converge rapidly; however, their computational complexity is greater than the existing model conversion algorithms. As a result, the proposed algorithms may not necessarily give the fast computational time. The computational time often depends upon the convergence speed and the computational complexity of the algorithm. The computational time due to the computational complexity can be improved by using advanced computer technologies such as the parallel computational techniques etc.

Acknowledgements—This work was supported in part by the US Army Research Office, under Research Contract DALL-03-87-K0001, the US Army Missile R & D Command, under Contract DAAH 01-85-CA111, and NASA–Johnson Space Center, under Contract NAG 9-211.

REFERENCES

1. N. K. Sinha and G. J. Lastman, Transformation algorithm for identification of continuous-time multivariable systems from discrete data, *Electron. Lett.* **17,** 779–780 (1981).
2. N. K. Sinha and B. Kuszta, Modeling and Identification of Dynamic Systems, pp. 98–102. Van Nostrand-Reinhold, New York (1983).
3. S. S. Haykin, A unified treatment of recursive digital filtering, *IEEE Trans. Automat. Control* **AC-17,** 113–116 (1972).
4. N. K. Sinha and S. Puthenpura, Choice of the sampling interval for the identification of continuous-time systems from samples of input/output data, *IEE Proc.* **132,** 263–267 (Pt.D) (1985).
5. M. Jamshidi, *Large-scale Systems: Modeling and Control.* North-Holland, New York (1983).
6. L. S. Shieh, Y. F. Chang and R. E. Yates, A dominant-data matching method for digital control systems modeling and design, *IEEE Trans. ind. Electr. Control Instrum* **IECI-28,** 390–396 (1981).
7. L. S. Shieh, Y. F. Chang and R. E. Yates, Model simplification and digital design of multivariable sampled-data control systems via a dominant-data matching method. *Appl. Math. Modelling* **8,** 355–364 (1984).
8. C. Moler and C. Van Loan, Nineteen dubious ways to compute the exponential of a matrix. *SIAM Rev.* **20,** 801–836 (1978).
9. E. L. Harris, Using discrete models with continuous design packages. *Automatica* **15,** 97–99 (1979).
10. L. S. Shieh, H. Wang and R. E. Yates, Discrete-continuous model conversion. *Appl. Math. Modeling* **4,** 449–455 (1980).
11. N. K. Sinha and G. J. Lastman, Identification of continuous-time multivariable systems from sampled data. *Int. J. Control* **35,** 117–126 (1982).
12. L. S. Shieh, J. S. H. Tsai and S. R. Lian, Determining continuous-time state equations from discrete-time state equations via the principal qth root method. *IEEE Trans. Automat. Control* **AC-31,** 454–457 (1986).
13. M. L. Liou, A novel method of evaluating transient response. *Proc. IEEE* **54,** 20–23 (1966).
14. J. B. Mankin and J. C. Hung, On round-off errors in computation of transition matrices. *Proc. JACC,* pp. 60–64 (1969).
15. L. S. Shieh, Y. T. Tsay and R. E. Yates, Computation of the principal nth roots of complex matrices. *IEEE Trans. Automat. Control* **AC-30,** 606–608 (1985).
16. Y. T. Tsay, L. S. Shieh and J. S. H. Tsai, A fast method for computing the principal nth roots of complex matrices. *Linear Algebra Applic.* **76,** 205–221 (1986).
17. N. J. Higham, Newton's method for the matrix square root. *Maths Comput.* **46,** 537–549 (1986).
18. L. S. Shieh, Y. T. Tsay and R. E. Yates, Some properties of matrix sign functions derived from continued fractions. *IEE Proc.* **130** (3), 111–118, Pt(D) (May 1983).
19. P. D. McMorran, Design of gas-turbine controller using inverse Nyquist method. *IEE Proc.* **117,** 2050–2056 (1970).
20. A. George, G. W. Stewart and R. Voigt, A special volume of linear algebra and its applications on parallel computing. *Linear Algebra Applic.* **77,** 1–345 (May 1986).
21. G. O. Beale and G. Cook, Frequency domain synthesis of discrete representations. *IEEE Trans. Ind. Elect. Control Instrum* **IECI-23,** 438–443 (1976).

APPENDIX A

Proof of the Numerical Stability of the Algorithm in Equations (21)

Consider the algorithm in equations (23) with $q = 1$ and $M(k) \triangleq \exp(-A(k)T)G$, i.e.

$$M(k+1) = M(k)\exp[I_n - M(k)], \quad M(0) = G \tag{A.1a}$$

$$A(k+1) = A(k) - \frac{1}{T}[I_n - M(k)], \quad A(0) = 0_n \tag{A.1b}$$

$$\lim_{k \to \infty} M(k) = I_n \tag{A.1c}$$

$$\lim_{k \to \infty} A(k) = A. \tag{A.1d}$$

It is desired to show that the iteration in equations (A.1) is numerically stable in the sense that a small perturbation introduced at the kth iteration due to round-off errors cannot lead to unbounded perturbations on succeeding iterates.

Let the perturbed models be $\tilde{M}(k)$ and $\tilde{A}(k)$ and the associated round-off errors be $E(k)$ and $F(k)$, respectively. Hence

$$\tilde{M}(k) = M(k) + E(k) \tag{A.2a}$$

$$\tilde{A}(k)T = A(k)T + F(k). \tag{A.2b}$$

Our aim is to analyze how the error matrices $E(k)$ and $F(k)$ propagate at the $(k+1)$ stage. To simplify the analysis we assume that no round-off errors occur when we compute $\tilde{M}(k+1)$ and $\tilde{A}(k+1)$ in the following:

$$\tilde{M}(k+1) = \tilde{M}(k)\exp[I_n - \tilde{M}(k)] \tag{A.3a}$$

$$\tilde{A}(k+1)T = \tilde{A}(k)T - [I_n - \tilde{M}(k)]. \tag{A.3b}$$

Substituting equations (A.2a) into (A.3a), expanding the matrix exponential function in equation (A.3a) and omitting the high order trivial terms of $E(k)$ we have

$$\begin{aligned} E(k+1) &= \tilde{M}(k+1) - M(k+1) = [M(k) + E(k)]\{I_n + [I_n - M(k) - E(k)] \\ &\quad + \tfrac{1}{2}[(I_n - M(k))^2 - (I_n - M(k))E(k) - E(k)(I_n - M(k))] + \cdots\} - M(k)\exp[I_n - M(k)] \\ &= M(k)[I_n + (I_n - M(k)) + \tfrac{1}{2}(I_n - M(k))^2 + \cdots] \\ &\quad + E(k)[I_n + (I_n - M(k)) + \tfrac{1}{2}(I_n - M(k))^2 + \cdots] \\ &\quad - M(k)\{E(k) + \tfrac{1}{2}[(I_n - M(k))E(k) + E(k)(I_n - M(k))] + \cdots\} - M(k)\exp[I_n - M(k)] \\ &= E(k)\exp[I_n - M(k)] - M(k)\{E(k) + \tfrac{1}{2}[(I_n - M(k))E(k) + E(k)(I_n - M(k))] + \ldots\} \end{aligned} \tag{A.4a}$$

Also, from equations (A.1b), (A.2b) and (A.3b) we obtain

$$F(k+1) = \tilde{A}(k+1)T - A(k+1)T = \tilde{A}(k)T - I_n + \tilde{M}(k) - A(k)T + I_n - M(k) = F(k) + E(k). \tag{A.4b}$$

When $k \to \infty$, $\exp(-A(k)T) \to G^{-1}$, hence $M(k) = \exp(-A(k)T)G \to I_n$. As a result, $\exp[I_n - M(k)] \to I_n$. Thus, equations (A.4) becomes

$$E(k+1) = 0_n \tag{A.5a}$$

$$F(k+1) = F(k) + E(k). \tag{A.5b}$$

The discrete-time state equation in equation (A.5b) with an identity system matrix is stable. If we make a further assumption that no round-off errors are introduced at the $(k+2)$ stage of the iteration, then equation (A.5b) becomes $F(k+2) = F(k+1) + E(k+1) = F(k+1)$. This suggests that the perturbation introduced at the kth stage has only a bounded effect on succeeding iterates. Therefore, the algorithm in equations (A.1) is numerically stable. In a similar manner, we can show that the algorithms in equations (24), (25) and (26) with $q > 1$ are numerically stable. Note that $M(k) = \exp(-A(k)T)G \simeq G^{-1}G = I_n$, even if the given matrix G is a stiff matrix.

APPENDIX B

Proof of the Numerical Stability of the Algorithm in Equations (31)

Consider the algorithm in equations (33) with $q = 1$ and $N(k) \triangleq AT - \ln(G(k))$, i.e.

$$N(k+1) = N(k) - \ln[I_n + N(k)], \quad N(0) = AT \tag{B.1a}$$

$$G(k+1) = G(k)[I_n + N(k)], \quad G(0) = I_n \tag{B.1b}$$

$$\lim_{k \to \infty} N(k) = 0_n \tag{B.1c}$$

$$\lim_{k \to \infty} G(k) = \exp(AT) = G. \tag{B.1d}$$

It is desired to show that the iteration in equations (B.1) is numerically stable in the sense that a small perturbation at the kth iterate due to round-off errors cannot lead to unbounded perturbations on succeeding iterates.

Let the perturbed models be $\tilde{N}(k)$ and $\tilde{G}(k)$ and the associated round-off errors be $E(k)$ and $F(k)$, respectively. Hence

$$\tilde{N}(k) = N(k) + E(k) \tag{B.2a}$$

$$\tilde{G}(k) = G(k) + F(k). \tag{B.2b}$$

Our purpose is to analyze how the error matrices $E(k)$ and $F(k)$ propagate at the $(k+1)$th stage. To simplify the analysis we assume that no new round-off errors occur when we compute $\tilde{N}(k+1)$ and $\tilde{G}(k+1)$ in the following:

$$\tilde{N}(k+1) = \tilde{N}(k) - \ln[I_n + \tilde{N}(k)] \tag{B.3a}$$

$$\tilde{G}(k+1) = \tilde{G}(k)[I_n + \tilde{N}(k)]. \tag{B.3b}$$

Substituting equations (B.2) into (B.3), expanding the matrix logarithmic function in equation (B.3a) and omitting the high order trivial terms of $E(k)$ yield

$$E(k+1) = E(k) - \{E(k) - \tfrac{1}{2}[N(k)E(k) + E(k)N(k)] + \tfrac{1}{3}[N(k)^2E(k) + N(k)E(k)N(k) + E(k)N(k)^2] - \cdots\} \tag{B.4a}$$

and

$$F(k+1) = F(k) + G(k)E(k) + F(k)N(k). \tag{B.4b}$$

When $k \to \infty$, $G(k) \to \exp(AT)$, hence $N(k) = AT - \ln(G(k)) = AT - \ln(\exp(AT)) \to 0_n$. Thus, equations (B.4) becomes

$$E(k+1) = 0_n \tag{B.5a}$$

$$F(k+1) = F(k) + G(k)E(k). \tag{B.5b}$$

The discrete-time state equation in (B.5b) with an identity system matrix is stable. If we make a further assumption that no new round-off errors are introduced at the $(k+2)$ stage of the iteration, then equation (B.5b) becomes $F(k+2) = F(k+1) + G(k+1)E(k+1) = F(k+1)$. This suggests that the perturbations introduced at the kth stage have only a bounded effect on succeeding iterates. Thus, the algorithm in equations (B.1) is numerically stable. In a similar manner, we can prove that the algorithms in equations (32), (33) and (34) with $q > 1$ are numerically stable. Since $N(k) = AT - \ln(G(k)) \simeq AT - AT = 0_n$, the given matrix A can be a stiff matrix.